Semantic super-resolution: when and where is it useful?

Radu Timofte[a], Vincent De Smet[b], Luc Van Gool[a,b]

[a]*ETH Zurich, D-ITET, Computer Vision Lab, Switzerland*
[b]*KU Leuven, ESAT - PSI, VISICS, Belgium*

Abstract

Recent algorithms for exemplar-based single image super-resolution have shown impressive results, mainly due to well-chosen priors and recently also due to more accurate blur kernels. Some methods exploit clustering of patches, local gradients or some context information. However, to the best of our knowledge, there is no literature studying the benefits of using semantic information at the image level. By semantic information we mean image segments with corresponding categorical labels. In this paper we investigate the use of semantic information in conjunction with A+, a state-of-the-art super-resolution method. We conduct experiments on large standard datasets of natural images with semantic annotations, and discuss the benefits vs. the drawbacks of using semantic information. Experimental results show that our semantic driven super-resolution can significantly improve over the original settings.

Keywords: single-image super-resolution, image enhancement, semantic segmentations, sparse coding

1. Introduction

Single-image super-resolution (SISR) is a branch of image enhancement that tries to add high-frequency information to low-frequency images in order to improve their sharpness during upsampling. Because SISR is an ill-posed problem these algorithms use different kinds of image priors to guide the creation of a high-resolution (HR) output. One popular class of priors assumes continuity between intensity values of neighboring pixels. They encourage the algorithm to find solutions that have as little change in neighboring intensity values as possible while still being faithful to the low-resolution (LR) input. This tendency of slow spatial variation is also observed in natural images. Some SISR algorithms get their priors from a database of image examples [1, 2, 3, 4, 5, 6]. These algorithms are trained on a collection of natural images in which (by down- and upsampling) they find numerous examples of corresponding local LR/HR combinations, either by working with small intensity patches [1] or by analyzing gradients [7].

In this paper we would like to highlight an often overlooked problem that comes with the ill-posed nature of the SR process. The transformation from an HR image patch to an LR image patch brings with it an inherent loss of information, meaning that a large number of different possible HR patches, when downsampled to LR space, are transformed onto one single LR point. This brings with it a certain ambiguity concerning the inverse transformation from LR to HR. Some algorithms simplify this and assume that finding the best matching LR patch in the database will result in the retrieval of the best available HR patch. Others (*e.g*. Freeman *et al*. [1]) try to enforce continuity along neighboring HR patches by choosing the proper candidate from k nearest neighbors instead of the closest match. This however gives no guarantee of a more correct solution, but rather a self-consistent one.

When we only look at local features, such as the ones depicted in Fig. 1, we have a difficult time estimating what the HR versions should look like exactly. We can make a

Email addresses: radu.timofte@vision.ee.ethz.ch (Radu Timofte), irvincentdesmet@gmail.com (Vincent De Smet), luc.vangool@esat.kuleuven.be (Luc Van Gool)

guess, *e.g*. by saying that edges should look sharper, but this still leaves room for many different solutions. If we then add semantic information about the scene by looking at a zoomed out version of the image (*e.g*. the top image in Fig 1) we can make a much more educated guess. In the case on top: the local feature is part of a car, more specifically the right tail light. Using this information we should be able to get a better HR patch. If however the zoomed out image is the one at the bottom of Fig 1, then we should reach an entirely different HR patch depicting an eye. Antonio Torralba [8] shows some very interesting examples of this problem for object detection. His example of a vague 'blob' in a blurry image can be interpreted as different things depending on the context in which it is shown (in the street, inside) or on its orientation (horizontal: car, vertical: person crossing the street).

In the following sections we will analyze when and where using semantic context information can help super-resolution most, and how this information can be incorporated as an extra prior in standard methods with minimum changes. Deriving substantially new methods or sophisticated methods for certain semantic contexts is out of the scope of our study. Moreover, we assume that the semantic information is known at pixel-level for most of our experiments. We then also evaluate the framework under real conditions, when the semantic information is less accurate. We have a number of hypotheses to check.

1. How can we adapt current super-resolution methods to use semantic information?
2. How can semantic information push the theoretical limits of what super-resolution can do?
3. How can semantic information help super-resolution in practice?
4. When is using semantic information desirable?

2. Related work

The idea of using priors specifically designed for certain types of textures or semantic classes has mainly been investigated implicitly. In order to test a certain SISR method, authors often tend to use similar images for training and testing (*e.g*. images containing text) [9]. With progress in automated object class detection comes the possibility to flexibly adapt priors across the image. Our paper investigates the promise this holds.

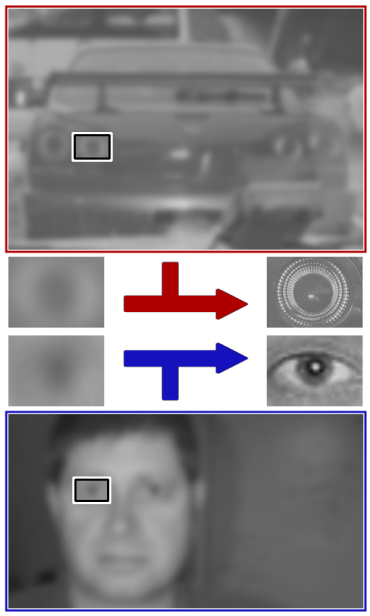


Figure 1: Semantic information can help guide the super-resolution process. This illustrative example shows two very similar low-resolution local features with their two corresponding high-resolution ground truth patches. The two full scale images show the blurry input images from which the features were taken. Depending on the context we should super-resolve these patches differently.

Some research has been done regarding the use of texture-dependent SR. HaCohen *et al*. [10] suggest a manual segmentation of the input image to define regions of homogenous texture. These regions are then super-resolved using a texture-specific prior based on well-chosen example images of each texture. They achieve good results but require very similar high-res versions of the textures in question for their examples. Tai *et al*. [11] suggest a similar approach in which the user provides an image that contains the same textures as the input image, and then use a combination of edge-based and learning-based SR. Tappen *et al*. [12] use example images to super-resolve faces by finding the same person in a database and warping their face to match the input image so they can use very local priors. Their results depend for a large part on finding the same person in the database, but also imply the benefits of using semantic information to super-resolve faces. Sun *et al*. [13] propose a method that uses a large training database of clustered LR/HR texture seg-

ments. An LR input image is then divided into texturally similar regions and for each region a similar cluster is found in the LR training database. These nearest neighbor textures are used as a prior for SR. This approach differs from our research in two ways. 1) This concerns texture regions specifically, not semantic regions like those corresponding to an object. It can only be used for regions with consistent textures, and will not necessarily produce good results for semantic regions which do not show large areas of similar texture, *e.g*. faces. 2) This method depends on texture matching in LR space, which does not guarantee similarity in HR space. This works for some textures, but not all. Using semantic information seems more suitable. In the direction of Sun *et al*. [13], Purkait and Chanda [14] partition the natural images into 'documents' (image regions) which are then grouped into 'topics' using probabilistic Latent Semantic Analysis (pLSA) [15]. For each topic specific sparse coding dictionaries are learned to then apply them locally at test.

Another related approach is the one presented by Sun and Hays [16], where the authors use scene descriptor features on an 'Internet-scale' image collection of more than six million images to find similar scenes to the LR input image, which are then used as a prior. This approach is highly data-driven and requires the presence of a similar image to the LR input image in the database, but can achieve good results when such an image is found. It still mixes LR/HR patch combinations coming from different types of scene segments though.

We would like to end this section by noting that SR methods focusing on patch recurrence across scales, like the influential work of Glasner *et al*. [17] and more recently Michaeli *et al*. [18] and Yang *et al*. [19], implicitly use semantic information as they use a prior based on the fact that most patches in an image have very similar patches somewhere in a downsampled version of the image itself. These methods ([17, 18, 19, 20]) only use the input image as a patch dataset and thus they tend to use patches from semantically identical content, although this is still not guaranteed. These methods are known to produce surprisingly good results, which might be partially attributed to this semantic consistency.

3. Benchmarking

The main drawback of using semantic information is the need for a pixel-level semantic label. This can be provided manually or automatically as the output of segmentation procedures such as Ladicky *et al*. [21] or any specific object detectors (*e.g*. faces, cars, people) such as Sadeghi and Forsyth [22] or Benenson *et al*. [23]. We will examine the effects of using semantic information for super-resolution on different datasets for which we have ground truth semantic labellings for both training and testing, using an upscaling factor of $\times 3$. One can expect the advantages to be dependent on the visual similarity between the various semantic classes involved. Outside of this inter-class patch variance we can also expect that the intra-class patch variance has an effect on the possible benefits. Classes with very low intra-class variance may benefit more from specialized trained dictionaries. Therefore we will examine three datasets that exhibit very different class properties.

3.1. KITTI dataset

The KITTI Vision Benchmark Suite [24] is a collection of annotated data from rural areas and highways, captured by equipping a station wagon with a stereo camera rig. The dataset was designed to develop challenging real-world benchmarks for different applications like stereo, optical flow, visual odometry, 3D object detection or tracking.

We use 30 training images and 30 testing images from KITTI as annotated by Ladicky *et al*. [21]. While originally the dataset uses 13 class labels, we will only work with 8 of these classes, because the remaining ones occur only seldom in the images.

The variance inside each semantic class is considerably high, with the notable exception of the 'sky' class. For some classes physically and visually distinct objects are grouped under the same class label at several different scales, locations and times.

3.2. Faces dataset

The second dataset is part of the Caltech 101 dataset [25] commonly used for object recognition. We use 10 training images and 11 testing images from the 'Faces' object category. We work with 2 semantic labels, the 'face' (foreground) and the 'non-face' (background).

The 'face' exhibits a fairly low intravariance with respect to the highly variant background.

3.3. License plates dataset

Lastly we will experiment on a dataset focused on license plates and cars. We use the license plate dataset proposed in [26] because of its large selection of 470 annotated/cropped license plates and 40 car images. 30 images are used as test images and the other for training. Again we have 2 semantic labels, 'license plate' and 'background'. This is an example of a dataset where we have a class ('license plate') with very low intravariance and another class with high intravariance ('background'). There is relatively little overlap between the appearance of these two classes, meaning we have high inter-class variance in this dataset.

4. Case Study 1: Nearest exemplar (NN)

The core of exemplar-based super-resolution lies in the assumption that for a known LR patch, its corresponding unknown HR patch can be found in a pool of training samples by searching for similar LR patches and retrieving their corresponding HR patches. The chances of success are good whenever we have a very large or limitless pool of LR/HR exemplars. On the downside, the larger the pool, the larger the incidence of cases in which almost identical LR patches/features have significantly different corresponding HR patches.

We define the following basic super-resolution method, named here NN (nearest neighbor based super-resolution). We use the same LR features as Zeyde *et al*. [4] and we use normalized HR patches by subtracting the bicubic interpolation patch to construct the training pool of exemplars. Thus, for LR features we have a set of 4 filter responses over the bicubic interpolated LR image (horizontal/vertical first and second order gradients). Through PCA the dimensionality of the features is reduced such that 99.9% of its energy is preserved (resulting in about 30 dimensions for upscaling factor $\times 3$).

For a given LR input patch this NN method will simply find the nearest neighbor (*i.e*. at the shortest Euclidean distance) in the pool of LR patches and inherit the corresponding HR patch to construct the super-resolved output. We work on a set of LR patches extracted over a grid. The HR output image pixels are gathered as the average of the pixels from the overlapping HR patches proposed during the LR matching process. The final output image is formed by adding the bicubic interpolation of the LR image to the reconstructed HR part.

4.1. Adding semantics

In a semantic setup each pixel has a known semantic label assigned to it. Correspondingly, each patch has a known semantic label. Here we assign to a patch the label of its center pixel. The 'global' pool of exemplars can be seen as the union of the pools corresponding to each semantic label. The semantic variant of NN will use the semantic label to bound the search of the nearest neighbor to the pool sharing the same label with the input LR patch.

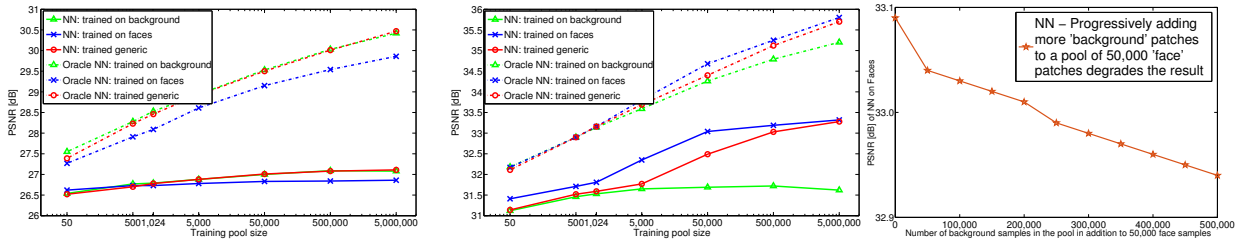
4.2. Oracle NN

Based on the NN algorithm described above, we derive an 'Oracle NN' variant in order to see the theoretical upper bound performance of NN. In the oracle setting we have access to the HR ground truth patches and simply replace them with their closest HR exemplars from the training pool. In terms of the algorithm, instead of using the LR patch features for the nearest neighbor search, we use the corresponding ground truth HR patches to find the most similar HR database patches. The oracle method basically avoids the ambiguity present in the original method, *i.e*. that identical looking LR patches can have very different HR counterparts. Therefore, the oracle NN method retrieves the most favorable HR patch from the pool.

4.3. Experiments

NN is a simple patch-based super-resolution method, and its simplicity allows us a better understanding of its limitations. Based on NN and Oracle NN, we first investigate the impact of the pool size, then the impact of using the semantic labels and moreover the correlation between the semantic pools, that is, how the super-resolution performance is affected if we use a pool from a different semantic label than the input LR patch. For this purpose we use the Faces dataset.

We conduct experiments with training pool sizes of 50 up to 5 million samples. We use the semantic labels to



a) Tested on 'background' b) Tested on 'faces'

Figure 2: NN performance on Faces dataset vs. training pool size and the semantic class.

train for 'Faces' or for 'Background' and to test the NN and Oracle NN methods with training pools from either the 'Faces' or the 'Background' semantic class. We also test with a 'Generic' training pool consisting of patches from both classes. In Fig. 2 we plot the experimental results.

4.3.1. Pool size

The larger the pool size the better the PSNR performance of the NN method. This is explained by the increased chance to find samples in the pool that resemble the ones in the testing material. As shown in Fig. 2, the performance of NN tends to increase monotonically with respect to the sample pool size inside the tested range. Fig. 2 b) even shows an increase of 1.5dB when the pool is increased from 50 to 5 million samples for the 'faces' semantic class. The largest improvement in the performance is up to about 50000 samples, after which we need to add considerably more samples for any additional improvement. This is because at that point the samples already capture the patch distribution well. Moreover, while Oracle NN is improving at a higher rate and keeps improving beyond this point, NN is unable to solve the ambiguities inherent in the SR problem, especially when we train on 'background'. That is, when we have enough samples to find good matches in LR space, we run into the additional problem that a variety of different HR patches can have very similar LR counterparts. This results in 'background' trained NN reaching a plateau much sooner (and lower) than 'faces' trained NN. Training only on faces reduces the ambiguity and allows the performance to keep rising with the pool size until a much later point.

We can draw 2 conclusions here:

1. the larger the pool the better the performance, yet the running time increases (near)linearly with the pool size;
2. there is a large gap between the oracle performance and the normal NN performance and this increases with the pool size. When tested on 'background', as few as 50 samples are sufficient for Oracle NN to exceed the performance of NN with 5000000 samples (!). When tested on 'faces' (Fig. 2 b)), we can see that the performance of NN trained with only faces keeps rising well beyond the point where the more general 'background' trained NN starts reaching a plateau.

4.3.2. Semantic super resolution

We use NN and Oracle NN to study how beneficial it is to have a pool of training samples from the same semantic class as the testing samples. As shown in Fig. 2 b), when NN uses 50000 training samples from 'Faces' to super-resolve the same semantic class we have a clear benefit of 1.5dB over the case where NN uses 'Background' samples to super-resolve 'Faces'. The benefit is smaller for small pools and increases with the size of the training pool. The explanation is that whenever we have a match between the training pool of samples and the testing images, any additional samples are meaningful and help to further improve the performance. If the training pool comes from a different semantic class then we need considerably more samples such that the pool will contain the information needed to achieve a certain performance. The 'Generic' curve stays with the 'Background' curve until it contains enough 'Face' samples and eventually when we test on 5 million samples it reaches the same PSNR as the 'Faces' curve, because by then most matching patches come from 'Face' labeled pool patches. In the rightmost

graph in Fig. 2 b) we show that performance drops for 'Faces' if we add background patches to a fixed pool of 50000 'Faces' patches. If instead of testing on 'Faces' we test on 'Background' we have a similar situation, as shown in Fig. 2 a). NN achieves better performance when the pool samples are from 'Background' in this case and benefits more from the additional samples than if we use the 'Faces' pool. When NN uses 'Faces' samples, the new samples cover the low-variance 'Faces' class but do not match as well to the higher-variance 'Background' class (being tested on), so the improvement we do get then is more from 'getting more and closer LR matches' than from 'getting the *right* LR matches' (which have more relevant HR patches).

The oracle bounds are quite meaningful for our two classes. When we super-resolve 'Faces' regardless of the origin of the samples in the pool, we have comparable oracle bounds for a pool size of up to 1000 samples, while when we super-resolve 'Background' there is a clear difference between using 'Faces' and 'Background' samples from the beginning. The 'Faces' samples are not sufficiently covering the 'Background' semantic space.

From our experiment we can conclude that:

1. a semantic match between the training samples and the testing samples is beneficial (especially for specific, low-variance semantic classes, such as 'Faces');
2. otherwise, we need a considerably larger pool of samples to achieve the same performance (*i.e*. for 'Faces' with 500 'Face' samples versus 500000 'Background' samples). Fig. 2 b) also suggests that the performance with 'Background' samples converges, meaning that adding samples may not be enough to reach the performance we get by using 50000 'Face' samples;
3. if the semantic information is unknown at test time, then it is more beneficial to have a large pool from a class with high variance or from a mix of diverse classes (*i.e*. 'Backgrounds'), this will assure a reasonable performance (*i.e*. the Oracle NN results are better when using 'Generic' samples than 'Face' samples in Fig. 2 a)).

5. Case Study 2: Sparse coding (Zeyde et al)

Exemplar-based super-resolution methods such as the one described in the previous section suffer from two main drawbacks: i) the demand for large pools for better performance, and ii) the ill-posed problem, one LR patch can have multiple corresponding HR patches in the pool. Zeyde *et al*. [4] propose an efficient alternative. It learns a reasonably small dictionary of (LR,HR) exemplars with large generalization property which tempers the main drawback (i) especially at test time. During the training the ill-posed problem (ii) is approached by assuming that if an LR patch has a sparse linear decomposition over the pool of LR patch features then its HR patch can be generated by imposing the same coefficients from the LR decomposition to the corresponding HR patches. Thus, the Zeyde *et al*. method tries to address both drawbacks. It is the most efficient pure sparse coding method, improving upon the seminal work of Yang *et al*. [3]. Note that the ill-posed problem is not completely solved here, it is just mitigated by averaging out contributions of different HR dictionary atoms. We chose to experiment with the method of Zeyde *et al*. because of its efficiency in both running time and qualitative performance with respect to the other sparse coding methods.

5.1. Oracle Zeyde

The sparsity idea behind Zeyde *et al*. is more complicated than the straightforward NN method. Instead of simply picking a convenient (LR,HR) pair of training patches now we mine for a sparse combination of pairs from the trained dictionary. However, we are still able to provide an oracle estimate for the Zeyde *et al*. approach. For the 'Oracle Zeyde' variant, instead of sparsely decomposing the LR patch feature over the LR dictionary, we directly decompose the HR ground truth patch over the HR patches of the dictionary and in the super-resolved output we use the resulting combination of HR dictionary patches. This sets an upper bound in the performance of the Zeyde *et al*. method.

5.2. Semantic super-resolution

As in the case of NN, for the method of Zeyde *et al*. we can employ the semantic information by separately training specialized dictionaries for each semantic label and

Table 1: Zeyde *et al*. PSNR results for upscaling $\times 3$ on KITTI dataset. Oracle Zeyde results are included in smaller print. We trained for each semantic label specialized dictionaries with 1024 atoms by using 0.5 million samples. We trained one 'generic' dictionary by ignoring the semantic labels and randomly sampling 0.5m patches. This is compared with a 'semantic' setting where we use the semantic label at test time to match with the specific trained dictionary. We also report the 'united' result where we join the semantic dictionaries together and do not use the semantic label at test. Additionally we report 'generic' with a 8192 dictionary. The color coding is as follows: we use blue to denote the results where we train and test on the same class, red for cases where training on another class gets (slightly) higher results, and green for the 'semantic' Zeyde results.

			Tested on								
Zeyde et al trained on	#atoms	Cod. time	sidewalk 0.7mil pxl	grass 0.2mil	car 1.5mil	fence 0.6mil	building 3.1mil	road 1.8mil	tree 4.8mil	sky 0.6mil	global 13.4mil
sidewalk	1024	3.2s	28.93/34.02	29.13/33.28	27.88/33.21	27.13/32.01	28.42/32.83	30.33/34.87	26.13/29.87	29.10/32.46	27.57/31.73
grass	1024	3.2s	28.82/33.16	29.12/32.63	27.72/32.31	27.12/31.16	28.27/31.94	30.20/34.19	26.12/29.29	28.98/31.81	27.50/31.04
car	1024	3.2s	28.94/34.09	29.15/33.40	27.98/33.65	27.16/32.36	28.51/33.37	30.36/35.05	26.15/30.08	29.16/32.79	27.62/32.02
fence	1024	3.2s	28.82/34.05	29.07/33.31	27.79/33.46	27.27/32.88	28.44/33.42	30.19/34.98	26.10/30.01	29.04/32.58	27.54/31.98
building	1024	3.2s	28.98/34.21	29.14/33.51	27.95/33.78	27.27/32.91	28.63/33.95	30.35/35.07	26.15/30.22	29.23/33.01	27.65/32.24
road	1024	3.2s	28.93/33.98	29.15/33.33	27.88/33.30	27.14/31.88	28.42/32.84	30.34/34.94	26.14/29.99	29.11/32.62	27.58/31.81
tree	1024	3.2s	28.74/32.87	29.11/32.66	27.68/32.20	27.08/31.55	28.32/32.41	30.11/33.88	26.14/29.48	29.05/32.16	27.51/32.22
sky	1024	3.2s	28.91/34.13	29.16/33.59	27.92/33.71	27.11/32.86	28.54/33.76	30.31/35.10	26.17/30.34	29.22/33.08	27.62/32.27
generic	1024	3.2s	28.93/34.03	29.14/33.37	27.91/33.51	27.21/32.54	28.52/33.51	30.33/34.97	26.16/30.05	29.17/32.79	27.61/32.01
semantic	8×1024	3.2s	28.93/34.03	29.12/32.68	27.98/33.62	27.27/32.78	28.63/33.89	30.34/34.95	26.15/29.53	29.24/32.91	27.65/31.71
united	8192	23s	29.03/35.05	29.21/34.21	28.06/34.72	27.34/33.71	28.68/34.65	30.41/35.80	26.21/30.93	29.30/33.69	27.71/32.97
generic	8192	23s	29.05/36.07	29.19/35.25	28.09/35.70	27.33/34.74	28.74/35/71	30.45/36.78	26.21/31.97	29.35/34.66	27.73/34.01

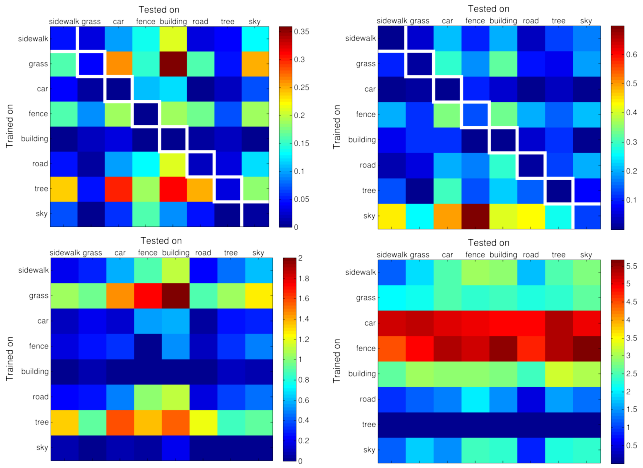
then for a new LR input applying the dictionary corresponding to the specific label of the LR input patch.

With 'Zeyde *et al*.' and 'Oracle Zeyde' we can now explore the role played by the semantic labels in the super-resolution performance. We use the KITTI dataset and train dictionaries using 0.5 million samples for each semantic class as extracted from the training images.

The results for Zeyde *et al*. are reported in Table 1. Fig. 3 a) shows a visual representation of these results. As expected after the NN experiments, for many semantic classes the best PSNR results are obtained when the method is trained using samples from the same semantic class. Nevertheless, there are classes such as 'building' with sufficiently large intra-variance that they form robust training dictionaries with good performance over all semantic labels and classes, as opposed to 'grass' that tends not to generalize well outside its own class. Also, we can notice large gaps between the best result usually achieved using training material from the same semantic class and the worst result, when there is an obvious mismatch between semantic classes. For example, on 'building' the best result is of 28.63dB and 0.36dB better than the 28.27dB result when we train on 'grass' class samples. From here we can conclude the importance of having a good match between the semantic information in the training material and the testing material. The 'semantic' Zeyde *et al*. setup uses the semantic class labels to pick the right dictionary for prediction. Overall, 'semantic' SR improves marginally over the 'generic' method (27.65dB vs. 27.61dB) when tested on the whole material ('global'). However, for some classes such as 'building' and 'car' the improvement is important, up to 1.1dB. So even when the global PSNR does not improve greatly, there can be large differences locally. Moreover, if we consider the semantic information known, 'semantic' has a time complexity equal with 'generic' despite using 8 times the amount of dictionaries (and training samples).

A bit disappointing is the fact that using a 'generic' $8\times$ larger dictionary or a 'united' dictionary of specialized semantic dictionaries, we achieve better PSNR results than the 'semantic' setup. Nevertheless, one should note that it comes at the expense of being $8\times$ slower, and that using larger dictionaries generally improves the super-resolution performance, as seen previously on the NN setup in Fig. 2.

From a theoretical point of view, looking at the Oracle Zeyde results in Table 1 is meaningful. Again, there is a good correlation between the oracle performance using the same semantic class samples for both training and testing. Oracle Zeyde trained on classes with high local redundancy such as 'sky' provides among the most robust results. This is explained by the fact that at the edges of the sky regions there are plenty of clean edge patches as well as textures, and the trained dictionary captures the most sensitive information well despite the redundant information.

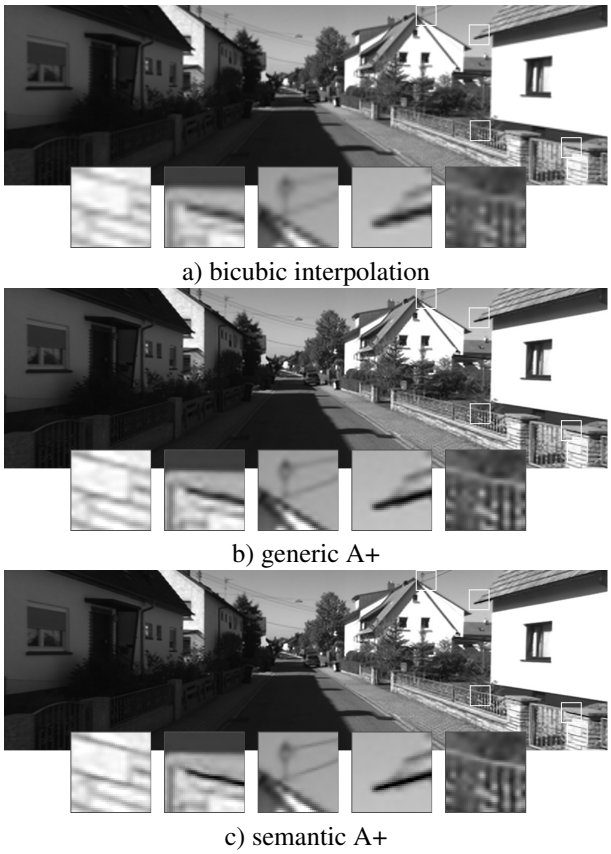


a) Zeyde b) A+

Figure 3: Top: Matrices with distances from the best PSNR for 8 KITTI classes. Each column shows which training class gives the best results for a specific tested class and how close other training classes get to this PSNR. Bottom: Same distance matrices for the Oracle algorithm versions.

6. Case Study 3: Neighbor embedding (A+)

The neighbor embedding based super-resolution methods assume a local LR manifold to be preserved in the local HR space. Chang *et al*. [2] were among the first to successfully introduce such a method. They employ Locally Linear Embedding [27] to perform a decomposition of the LR input patch over a local neighborhood of fixed size of LR training pool patches. They then impose the LR coefficients on the HR pool patches to reconstruct the corresponding HR patch. Timofte *et al*. [5] show that with their best respective settings, most neighbor embedding methods can reach comparable performance. They also propose to anchor the neighborhoods and to use ridge regression to learn local projections from the LR features space to HR space. This way each dictionary atom serves as an anchor that has its own stored neighborhood and projection matrix. Their Anchored Neighborhood Regression (ANR) method equals Zeyde *et al*. in performance but is computationally more efficient for the same dictionaries (ANR uses the dictionary samples as anchoring points). Recently, the same authors introduced A+ [6], which improves further on ANR. While ANR precomputes the local projections based on the neighborhoods from the dictionary itself, A+ uses the training pool of raw samples (which was used to form the dictionary). This improves



a) bicubic interpolation

b) generic A+

c) semantic A+

Figure 4: **KITTI class example:** semantic A+ provides clearer and sharper edges than generic A+ overall.

the qualitative performance beyond the results of Zeyde *et al*. and ANR. We therefore use A+ in our study as representative method for neighbor embedding based super-resolution.

6.1. Oracle A+

An Oracle A+ assumes that the assignment of the LR input patch to the anchoring point and regressor is ideal, therefore we use the ground truth HR patch of the input and pick the best anchoring point as the one with the highest HR correlation. Moreover, instead of using a regressor learned on the LR features of the training samples, we take their corresponding HR patches and derive on them the ideal regressor. Therefore, since we work on the HR part of the dictionary, we transform it first to an l_2-normalized basis. For the input LR patch, we take the

Table 2: A+ PSNR results for upscaling $\times 3$ on KITTI dataset. Same colors from Table 1.

A+ trained on	#atoms	#samples	Coding time	Tested on sidewalk 0.7mil pxl	grass 0.2mil	car 1.5mil	fence 0.6mil	building 3.1mil	road 1.8mil	tree 4.8mil	sky 0.6mil	global 13.4mil	global uniform
sidewalk	1024	0.5mil	1.3s	29.16	29.27	28.14	27.37	28.66	30.46	26.24	29.27	27.71	28.31
grass	1024	0.5mil	1.3s	29.07	29.30	28.06	27.29	28.54	30.41	26.30	29.25	27.69	28.19
car	1024	0.5mil	1.3s	29.17	29.30	28.35	27.37	28.81	30.50	26.33	29.43	27.81	28.39
fence	1024	0.5mil	1.3s	28.97	29.21	28.01	27.33	28.53	30.30	26.22	29.23	27.63	28.23
building	1024	0.5mil	1.3s	29.10	29.21	28.24	27.47	28.86	30.45	26.26	29.44	27.77	28.37
road	1024	0.5mil	1.3s	29.14	29.26	28.15	27.30	28.57	30.48	26.24	29.24	27.69	28.29
tree	1024	0.5mil	1.3s	29.03	29.32	28.05	27.33	28.64	30.35	26.37	29.36	27.75	28.31
sky	1024	0.5mil	1.3s	28.73	29.06	27.85	26.77	28.45	30.07	26.10	29.31	27.49	28.03
generic	1024	0.5mil	1.3s	29.11	29.29	28.19	27.41	28.75	30.43	26.32	29.38	27.77	28.35
semantic	8×1024	8×0.5mil	1.3s	29.16	29.31	28.35	27.35	28.86	30.49	26.37	29.47	27.84	28.42
united	8192	0.5mil	5.3s	29.14	29.33	28.22	27.43	28.76	30.48	26.32	29.41	27.78	28.37
semantic (automatic)	8×1024	8×0.5mil	1.3s	29.14	29.32	28.23	27.33	28.78	30.47	26.36	29.35	27.82	28.40
topic [14]	20 × 500	20000 docs	32s	28.50	28.80	27.52	26.89	28.21	29.86	25.93	28.99	27.33	27.86

a) generic A+

b) semantic A+

Figure 5: Another KITTI example. Using a specialized dictionary for the 'fence' class lets our semantic algorithm super-resolve the fence better, where the original A+ method averages out the details at the fence.

HR ground truth, find the most correlated HR anchoring point, and apply its ideal HR regressor to the HR ground truth. Based on the ideally HR reconstructed patches we achieve the Oracle A+ output, an upper bound in performance for the A+ method.

6.2. Semantic super-resolution

We follow the same steps here as for Zeyde *et al*. The semantic A+ variant uses trained dictionaries and corresponding regressors for each semantic label.

In the same manner as Zeyde *et al*. we evaluate A+ with and without semantic information on KITTI. The results are summarized in Table 2 and in Fig. 3. Fig. 3 shows the difference between each tested class and the one that has the best performance. As visually pointed out in Fig. 3 and 4, training on the same semantic category is beneficial for Zeyde and A+ – the best results are usually found on the diagonal where the test semantic category matches the trained semantic category. For the Oracle A+ results we get somehow unexpected results, with 'car' and 'fence' considerably less favorable to oracle results. A tuning of the training neighborhood size in A+ for these semantic classes would improve their regressors. However, Oracle A+ results are in the range of 45dB-53dB, and well above Oracle Zeyde. These results can be found in the supplementary material.

In Table 3 we report the A+ results on License plates dataset and Faces dataset. Notice that the semantic content in the dataset images is highly unbalanced – the number of 'background' pixels strongly dominates (50 to 1, on 'License plates'). This is the reason why the 'generic A+' highly correlates with the 'background' trained A+ results on the 'background' pixels. Therefore, we also report the PSNR results for a 'uniform' setting – the 'global uniform' results, where the contributions are equal for different semantic classes, corresponding to equal amounts of image pixels. The 'Semantic A+' PSNR improves over the 'generic A+' PSNR marginally on the global unbalanced evaluation (0.03-0.06dB), but if we analyze the improvement for specific foreground classes, the improvement is significant ($\sim$ 0.3dB), and moreover, if we report to the balanced 'global uniform' results again we observe consistent gains (0.27dB and 0.08dB). The PSNR improvement for 'faces' seems marginal mainly because the PSNR metric does not sufficiently capture perceptual quality. Faces consist of flat regions with sharp features at important locations. These flat regions stay the same for both results, while the features improve. The PSNR metric however does not reflect this perceived difference well.

Table 3: A+ PSNR results for upscaling $\times 3$ on License plate dataset and on Faces dataset.

trained on	#atoms	#samples	Coding time	A+ tested on License plates: foreground (plate) 0.1mil pxl	background 5.0mil	global 5.1mil	global uniform	A+ tested on Faces dataset: foreground (face) 0.4mil pxl	background 1.4mil	global 1.8mil	global uniform
foreground	1024	0.5mil	0.3s	23.62	30.19	29.90	25.76	32.53	27.13	27.82	29.04
background	1024	0.5mil	0.3s	22.78	30.50	30.10	25.11	32.09	27.29	27.93	29.06
generic	1024	0.5mil	0.3s	23.13	30.46	30.12	25.45	32.26	27.28	27.94	29.09
semantic	2×1024	2×0.5mil	0.3s	23.64	30.50	30.18	25.84	32.56	27.29	27.97	29.17
united	2048	0.5mil	0.5s	23.51	30.43	30.11	25.72	32.44	27.28	27.95	29.13
generic	2048	0.5mil	0.5s	23.38	30.47	30.13	25.61	32.27	27.29	27.95	29.10
semantic (automatic)	2×1024	2×0.5mil	0.3s	23.61	30.49	30.18	25.83	32.51	27.26	27.96	29.15
topic [14]	20 × 500	20000 docs	6.1s	22.52	29.95	29.58	24.87	31.73	26.99	27.63	28.77

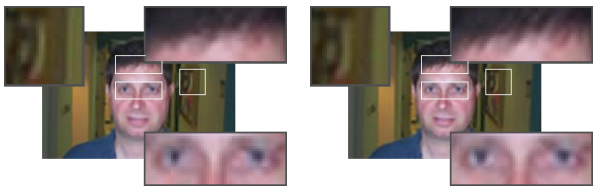
a) Query image: χ^2 probability = 1, F = 0.2126

b) License plate image: χ^2 probability = 0.3821, F = 0.2883

c) KITTI image: χ^2 probability = 0.3728, F = 0.2439

d) Face image: χ^2 probability = 0.0168, F = 0.4349

Figure 6: We compare three images (b), c) and d), taken from the three used datasets) to a query image (a)) using the χ^2 method described in Sec. 7. We also show the values of our proposed F feature for the three images and the query image, which shows how 'atypical' each image is compared to the others.



a) A+ trained on background

b) A+ trained on face

Figure 7: Face vs. background class example.

7. Dataset analysis

So when is it useful to use different dictionaries to super-resolve different semantic parts of an image? In some cases it is clearly favorable to train on a semantic subset of the training data. Tables 1-3 and Figs. 4-8 are examples of the quantitative and qualitative benefits of doing this. When two LR patch sources (from two different semantic classes) can be accurately modeled with the same probability density function, then the SR training phase will result in very similar dictionaries for both classes. On the other hand, when these probability density functions differ significantly the resulting dictionaries will be different and tuned for the specific content of each class.

One way to determine a distance between the probability density functions of two sample sources is the χ^2 distance. This however requires histogram analysis and our patch data is relatively high-dimensional (histograms of 81 dimensions (9×9 patches) require an unusably large amount of bins). There are multiple ways to get around this problem. One is to perform the histogram analysis not on the patches themselves, but on their proximity to the dictionary atoms. We can use a nearest neighbor search to find the closest atoms for a random subsampling of patches belonging to different classes. A histogram can be created from this for each class by counting how many patches 'belong' to each of the (*e.g.* 1024) dictionary atoms. This can then be used to calculate a χ^2 value, where the observed values are the histograms, and the expected values are taken from a histogram of general patches from all classes. Finally, from the χ^2 value we can calculate the probability that the patches come from the same pool. This is the approach we suggest for the KITTI dataset, which has 8 classes. For the other two datasets, 'license plates' and 'faces', we only have two classes. This means that, aside from comparing the classes to the general patch pool, we can also directly compare the two classes to each other, by using the histogram from one class as the observed variable, and the histogram from the other class as the expected variable. For these two classes we show both methods. We

summarise the results of these tests in Table 4. The most noticeable difference in the KITTI dataset can be seen in the 'sky' class, which has a low probability of having the same probability density function as the overall pool of patches. In the other datasets we can see that both the 'Faces' and 'License plates' classes are very dissimilar to the general pool of patches. We also show the probability for when we compare the two classes of *e.g.* the 'Faces' dataset to each other, rather than to the general pool of patches, as described above. This is shown in the second value. We can see that the two classes are in both datasets very likely coming from differing distributions.

In the previous paragraph we made use of the clustered patch dictionary of the ANR and A+ methods to calculate the probability of two patch sources sharing the same probability density function. However, not all super-resolution methods use this kind of clustering. We can propose instead to use a simple feature to give us an indication of the difference between two patch sources, independent of any explicit clustering: for a random subset of samples we find the nearest neighbors and check their labels. Do these nearest neighbor patches belong to the same class as the query patch? If for instance n patch sources have identical probability distributions then we expect the average amount of nearest neighbors belonging to the same class as the query (*e.g.* class i) to be $N_i/(N_1+N_2+...+N_n)$, with N_i the number of available samples belonging to class i.

We propose the feature

$$F^*(c_i) = \sum_{j=1}^{M} \frac{|P(c_i|x_j) - P(c_i)|}{M}, \qquad (1)$$

or the average distance between the local probability of class i given a certain input patch x_j of class i and the overall probability of class i, averaged over M tested patches. We can approximate $P(c_i|x_j)$ as the amount of local nearest neighbors of class i divided by the number of examined nearest neighbors and $P(c_i)$ as the ratio of class i patches to the entire database. We refer to this approximation as F.

If we apply this to our three tested datasets by taking an equal amount of random samples from each class (around 100000) we get the results in Table 5. The number of patches used is determined by the class with the lowest available amount of patches. When there are only two classes in a dataset then the F value is more or less symmetric. For the KITTI dataset, where we have 8 classes, there can also be relative differences between classes. The values in Table 5 reinforce the idea that (especially in the cases of faces and even more so for license plates) it is useful to model these different patch sources with separately trained dictionaries. The Faces and License plates datasets show a high response to our feature, indicating that these classes could benefit significantly from using separate dictionaries. For the KITTI dataset the 'sky' class gets the highest response, which in this case means that treating it separately could help the other classes by not overtraining on the highly redundant 'sky' patches.

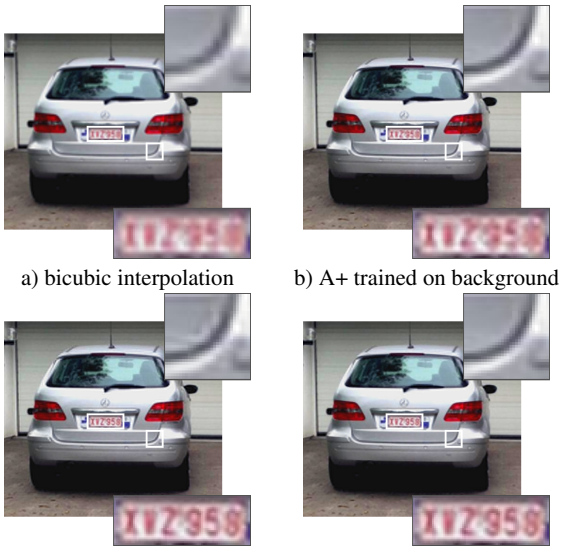
We visualize these two measures in Fig. 6. In this figure we show three images (one from each of our three datasets) and a query image with a car (shown on the left). We can use the probability from the χ^2 distance to measure how similar the content is. More specifically, we use the histogram of the query image (calculated with respect to a general dictionary of 1024 atoms) as a reference and the histograms from the three other images as the target. The probability of the query image is of course equal to 1, and we can see that images with similar content (depicting cars in this case) show a high probability of orig-

Table 4: χ^2 test performed on the three datasets by using the 1024 cluster centers from ANR and A+. For 'Faces' and 'License plates' we show two values, the first is the case where we compare to the general patch distribution, the same way the KITTI values are obtained. The second value is the probability when we calculate the χ^2 values comparing the class histograms to each other.

Dataset	class	F	Dataset	class	Probability of χ^2 value
KITTI	sidewalk	0.8411	Faces	background	0.8635/0.1135
	grass	0.8832		face	0.4961/0.3247
	car	0.8982			
	fence	0.8805			
	building	0.8898			
	road	0.8923	License plates	background	0.6338/0.0013
	tree	0.8930		plate	0.4507/0.1221
	sky	**0.3958**			

Table 5: The feature F indicates how separate a class is from the other classes in the dataset of patches.

Dataset	class	F	Dataset	class	F
KITTI	sidewalk	0.2100	Faces	background	0.3608
	grass	0.1806		face	0.3008
	car	0.2393			
	fence	0.2597			
	building	0.2582			
	road	0.2050	License plates	background	0.6047
	tree	0.1445		plate	0.5729
	sky	**0.4149**			



a) bicubic interpolation b) A+ trained on background

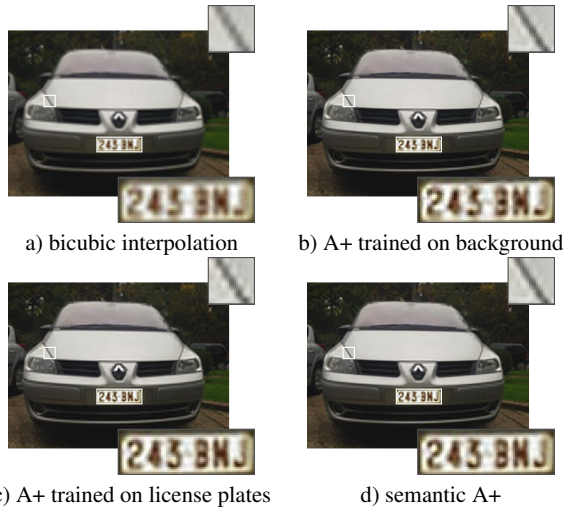
c) A+ trained on license plates d) semantic A+

Figure 8: License plate vs. background class example.

inating from the same semantic pool as the query image. The χ^2 probability for the 'Face' image is significantly lower than the others, indicating that this image has a low probability of belonging to the same group of images as the query image. We also show the F values for all the images in the figure. In this case the images are all compared to each other, and the F value can be interpreted as a measure for how 'atypical' each image is with respect to the pool of 4 images. The values are similar for all except the 'Face' image, where it is significantly higher. This is also what we expect, given that the content of that image (and thus also its image patches) is significantly different from the others.

8. Semantic super-resolution in practice

In all our previous experiments we considered that the semantic information is given and, moreover, ideally matches the ground truth at pixel-level. However, in practice, the automatic semantic labeling of an image at pixel-level is far from perfect. In this section we report results using A+ with and without semantic information and results for a method based on latent topics. For training we assume the pixel-level semantic labels to equal the ground truth manual annotations, while for testing we



a) bicubic interpolation b) A+ trained on background

c) A+ trained on license plates d) semantic A+

Figure 9: Another license plate vs. car/background class example.

work with the outputs of different automatic semantic segmentation methods. The semantic results are found in Table 2 for the KITTI dataset and in Table 3 for the Faces and License plates datasets under the label 'semantic (automatic)', while the default 'semantic' setup uses the ideal manual semantic information.

8.1. Automatic semantic labeling

As previously mentioned the advances in object detection provide a fairly cheap source of rough semantic segmentation of an image. Of course, for accurate semantics usually one needs larger image regions around the pixels than the local patches that are normally used in super-resolution algorithms. We directly employ in-house off-the-shelf face and license plate detectors [23] trained on extra data for our 'Faces' and 'License plates' datasets, and the automatic pixel-level semantic labeling method of Ladicky *et al*. [21] for the KITTI dataset. Unfortunately, the output of the face and license plate detectors is a bounding box, a rather rough segmentation of the object from its surroundings/background; while the automatic pixel-level labeling method used for the KITTI dataset is still far from perfect. In Fig. 13 we show the confusion matrices for our employed automatic labeling methods on each dataset. The pixel-level accuracy varies from 83.78% on KITTI to 94.12% on Faces and 99.04% on License plates dataset. A visual inspection shows typical labeling errors such as textureless surfaces or ambigu-

ous textures in the case of KITTI dataset (see Fig. 12), while in the case of Faces (see Fig. 10) and License plates (see Fig. 11) the errors are mainly due to the rough semantic segmentation. The accuracy of the automatic labeling method is very important. However, in particular cases, such as faces or license plates, one can afford working with rough segmentation (result of a detection process), as long as the region of interest is correctly identified and handled. Note that one can fully benefit from using the specific semantic information to further improve the performance in super-resolution. In this way, the faces can be aligned or warped, the scale, aspect ratio, or the relative position inside the aligned face can lead to refined super-resolution models. The same can be said about license plates. However, it is out of the scope of our paper to go in more refined models; we introduced general models and studied the impact of the labels and showed that usually the semantic information is beneficial for super-resolution.

8.2. Comparison with latent topics

In addition to our A+ derived setups, we report as reference the results of the method of Purkait and Chanda [14] based on probabilistic Latent Semantic Analysis (pLSA) [15]. The idea behind the method is to partition the natural images into 'documents' (regions) grouped to discover inherent 'topics' using pLSA for which sparse coding dual dictionaries are learned. At test the topic is inferred locally and the corresponding dictionary is applied. Most of the other related methods (see Section 2) are based on clustering of the patches (or contexts), and refined specific models. Note that A+ can be interpreted under the same paradigm since it partitions the data (patches) and for each partition (cluster of patches) stores specific anchors and regressors. For the topic method [14] we use the codes and the default settings as provided by the authors. Note that the method builds on top of the Yang *et al*. [3] sparse coding method. By adding the topic reasoning Purkait and Chanda [14] report an average improvement of 0.15dB on 8 images. In our experiments, we retrain the topic method for upscaling factor 3 for each dataset on the corresponding training images. Using the default factor and training images leads to slightly worse results. The results are reported for the KITTI dataset in Table 2 and for the Faces and License plates datasets in Table 3. Overall the topic method [14] is significantly computationally more demanding than the other A+ setups used and usually more than 0.3dB worse.

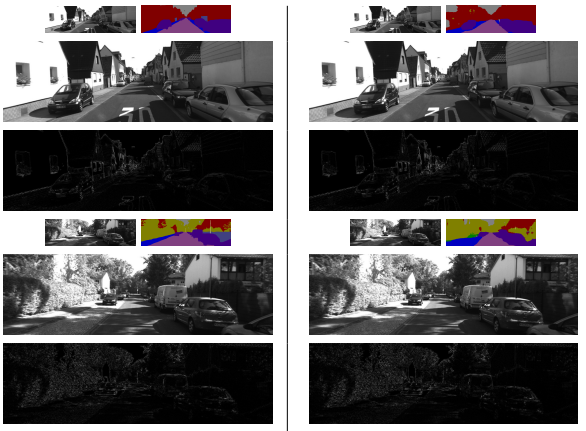


Figure 12: KITTI examples of semantic A+ results with manual or automatic labels and the corresponding pixels errors tripled for visualisation. The automatic labeling usually fails on textureless regions (top example: building labeled as sky) or when the same texture may belong to different semantic labels (bottom example: road stripe taken as sidewalk). Best seen on screen.

9. Conclusion

There has been little research into the use of semantic priors in super-resolution. Intuition however tells us that semantic context has a large influence on the appearance of objects, specifically in the high-res frequencies that need to be hallucinated by super-resolution algorithms. In this paper we proposed a framework to investigate the

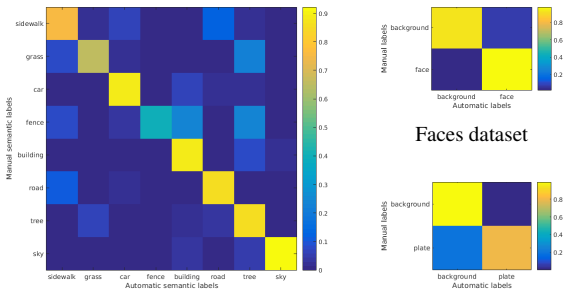


Figure 13: Confusion matrices in automatic semantic labeling. The automatic labeling has a pixel accuracy of 83.78% on KITTI dataset, 94.12% on Faces and 99.04% on License plates.

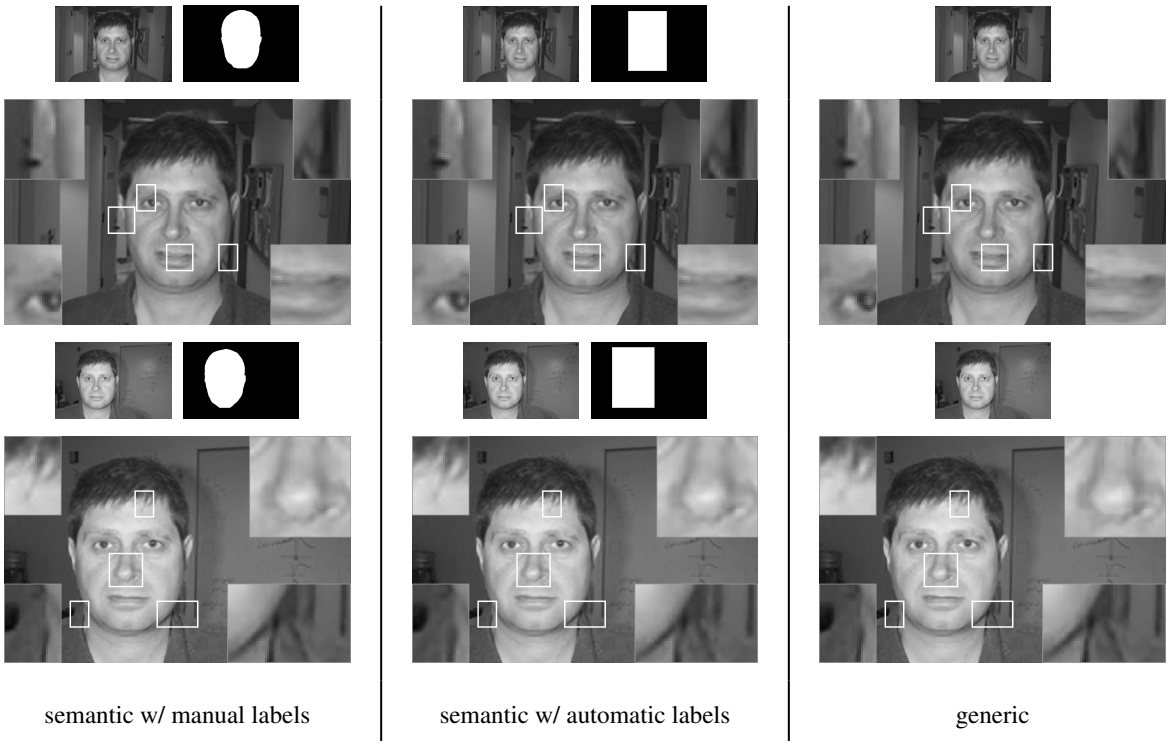


Figure 10: Face examples of results for semantic and generic A+ setups. The LR input and the labels are provided for each HR result ($\times 3$). Best seen on screen.

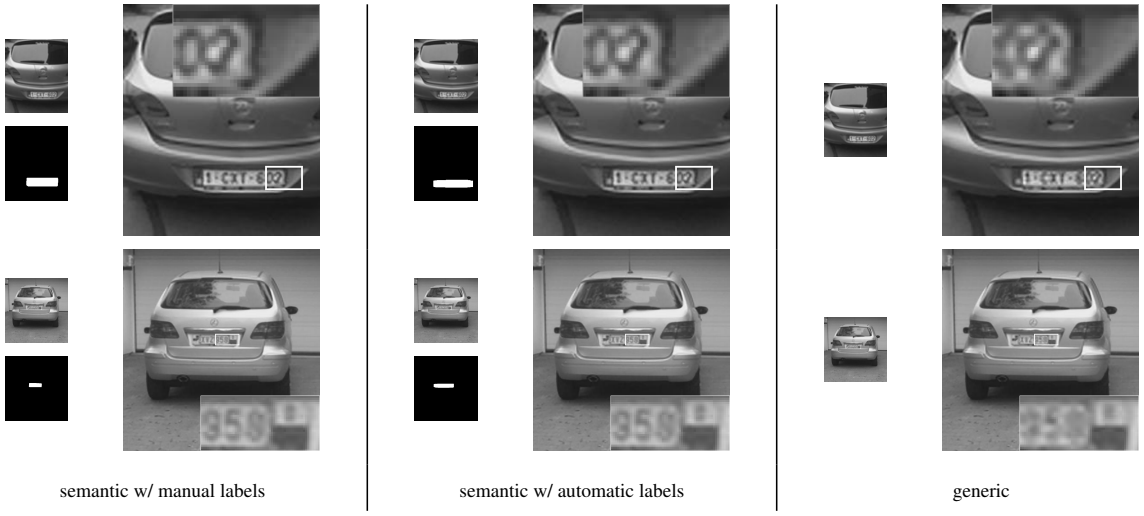


Figure 11: License plate examples of results for semantic and generic A+ setups. The LR input and the labels are provided for each HR result ($\times 3$). Best seen on screen.

possibilities of using semantic labels to improve the accuracy of single-image super-resolution results. We experimented with different state-of-the-art SR algorithms on the KITTI dataset, the Caltech101 faces dataset and a license plate dataset to show that using a semantic prior can have significant advantages. We also presented an oracle framework to explore the theoretical limitations of using semantic information and proposed a feature that indicates when it can be useful to separate the SR training for certain classes. Finally we test the framework on automatically segmented labelings, showing that for the datasets explored in this paper, the labels found from current segmentation algorithms combined with the robustness of the semantic SR framework can create super-resolved output images that reach very similar quality to the case where manual labels are used. We conclude that using specifically trained dictionaries can improve SR results significantly, and even in the cases where the global image PSNR is only slightly affected we can see significant local improvement for specific image segments.

Acknowledgments. The authors thank the reviewers for their valuable comments. This work was partly supported by the ETH General Founding (OK), ERC Advanced Grant VarCity (#273940), and the Flemish iMinds framework.

[1] W. T. Freeman, E. C. Pasztor, O. T. Carmichael, Learning low-level vision, International journal of computer vision 40 (1) (2000) 25–47.

[2] H. Chang, D.-Y. Yeung, Y. Xiong, Super-resolution through neighbor embedding, Computer Vision and Pattern Recognition (CVPR), 2004 IEEE Conference on 01 (2004) 275–282.

[3] J. Yang, J. Wright, T. S. Huang, Y. Ma, Image super-resolution as sparse representation of raw image patches, in: Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on, 2008, pp. 1–8.

[4] R. Zeyde, M. Elad, M. Protter, On single image scale-up using sparse-representations, in: Curves and Surfaces, 2010, pp. 711–730.

[5] R. Timofte, V. De Smet, L. Van Gool, Anchored neighborhood regression for fast example-based super resolution, in: Computer Vision (ICCV), 2013 IEEE International Conference on, 2013, pp. 1920–1927.

[6] R. Timofte, V. De Smet, L. Van Gool, A+: Adjusted anchored neighborhood regression for fast super-resolution, in: Asian Conference on Computer Vision (ACCV 2014), 2014. doi:10.1007/978-3-319-16817-3_8.

[7] J. Sun, Z. Xu, H.-Y. Shum, Image super-resolution using gradient profile prior., in: Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on, 2008, pp. 1–8.

[8] A. Torralba, Contextual priming for object detection, International Journal of Computer Vision 53 (2) (2003) 169–191.

[9] L. C. Pickup, S. J. Roberts, A. Zisserman, A sampled texture prior for image super-resolution, in: Advances in neural information processing systems, 2003, pp. 1587–1594.

[10] Y. HaCohen, R. Fattal, D. Lischinski, Image upsampling via texture hallucination, in: Computational Photography (ICCP), 2010 IEEE International Conference on, 2010, pp. 1–8.

[11] Y.-W. Tai, S. Liu, M. Brown, S. Lin, Super resolution using edge prior and single image detail synthesis, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 2400–2407.

[12] M. F. Tappen, C. Liu, A bayesian approach to alignment-based image hallucination, in: Computer Vision–ECCV 2012, 2012, pp. 236–249.

[13] J. Sun, J. Zhu, M. F. Tappen, Context-constrained hallucination for image super-resolution., in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 231–238.

[14] P. Purkait, B. Chanda, Image upscaling using multiple dictionaries of natural image patches, in: K. Lee, Y. Matsushita, J. Rehg, Z. Hu (Eds.), Computer Vision ACCV 2012, Vol. 7726 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 284–295.

[15] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, ACM, New York, NY, USA, 1999, pp. 50–57.

[16] L. Sun, J. Hays, Super-resolution from internet-scale scene matching, in: Computational Photography (ICCP), 2012 IEEE International Conference on, 2012, pp. 1–12.

[17] D. Glasner, S. Bagon, M. Irani, Super-resolution from a single image, in: Computer Vision, 2009 IEEE 12th International Conference on, 2009, pp. 349–356.

[18] T. Michaeli, M. Irani, Nonparametric blind super-resolution, in: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE, 2013, pp. 945–952.

[19] J. Yang, Z. Lin, S. Cohen, Fast image super-resolution based on in-place example regression, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, 2013, pp. 1059–1066.

[20] Y. Ogawa, Y. Ariki, T. Takiguchi, Super-resolution by gmm based conversion using self-reduction image, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, 2012, pp. 1285–1288.

[21] L. Ladicky, J. Shi, M. Pollefeys, Pulling things out of perspective, in: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, 2014, pp. 89–96.

[22] M. Sadeghi, D. Forsyth, 30hz object detection with dpm v5, in: Computer Vision–ECCV 2014, 2014, pp. 65–79.

[23] R. Benenson, M. Mathias, R. Timofte, L. Van Gool, Pedestrian detection at 100 frames per second, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012, pp. 2903–2910. doi:10.1109/CVPR.2012.6248017.

[24] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012, pp. 3354–3361.

[25] L. Fei-fei, R. Fergus, P. Perona, One-shot learning of object categories, Pattern Analysis and Machine Intelligence, IEEE Transactions on 28 (2006) 594–611.

[26] V. De Smet, V. P. Namboodiri, L. J. Van Gool, Nonuniform image patch exemplars for low level vision., in: Applications of Computer Vision (WACV), 2013 IEEE Workshop on, 2013, pp. 23–30.

[27] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.